

代码说明

一、需求背景

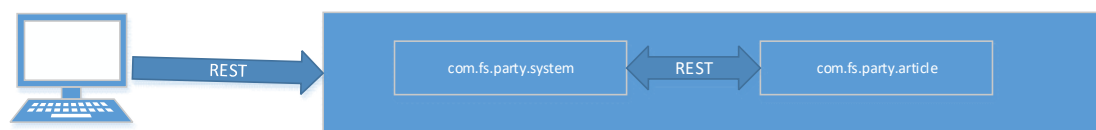
1.现在产品、项目迭代，更新需求越来越频繁，功能之间存在各种相同点和差异，如何能够做到代码的复用和隔离越来越成为一项重要的开发技能。

2.随着产品，项目周期的不断推进，功能越来越多，代码量越来越大，一个比较清晰的功能划分显得越来越重要，对于后期成员能够快速投入，我们需要一定的层次划分来降低学习成本。

二、设计思路



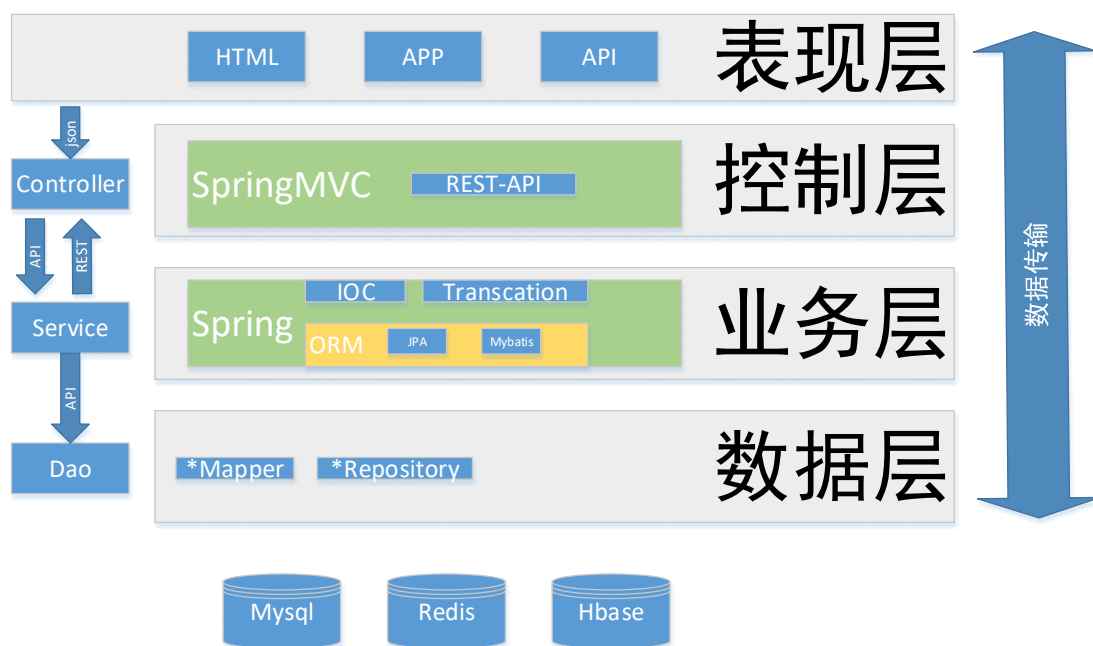
- `com-fs-party-starter-logging` 采用松耦合设计，避免业务模块的强依赖关系，在使用过程中，只需要添加到项目中就可以。
- `com.fs.party.depeds` 需要继承的基类，工具包，核心依赖项，基于 `pojo` 设计，与业务无关。
- `com.fs.party.article`、`com.fs.party.system` 业务功能项目，主要做业务数据过滤，存储处理业务。
- `com.fs.party.config.server` 配置中心服务，集中处理业务功能项目的各种配置场景。
- `com.fs.party.server` 服务注册中心，框架依赖功能。



- 客户端（PC、移动端）、业务之间等交互统一使用 `REST`，保证入口标准的一致。

三、实现说明

- 全局功能 (com.fs.party-starter-logging)
 - 拦截器
 - ◆ MyWebAppConfigurer 添加拦截器链，对项目全局配置项（如：项目跨域限制）
 - ◆ MyInterceptor（拦截器链中的一个实例），WebLogAspect 处理 WEB 调用时一些全局处理，如 WEB 参数日志，返回信息等。其中 WebLogAspect 功能更全面，可以自定义，以达到覆盖业务所有区域。
 - 参数日志
 - 等.....
- 工具包（继承类，工具类等）(com.fs.party.depends)
 - 业务日志注解 (OperationLog)
 - 基类 (controller,service,exception)
 - 全局常量
 - 数据处理工具类
 - 等.....
- 业务服务 (com.fs.party.article、com.fs.party.system)



- ◆ Controller 层 (controller 目录下)
 - WEB 入口，处理数据和跳转
- ◆ Service 层 (service 目录下)
 - 业务处理接口 (*Service)
 - 业务处理实现 (*ServiceImpl)
 - 业务项目间访问接口 (*Client)
 - 业务项目间访问实现及异常处理 (* ClientHystrix)
- 配置服务 (com.fs.party.config.server)
 - 略.....
- 注册服务 (com.fs.party.server)

■ 略.....

四、代码文件

1. *com.fs.party-starter-logging* 部分

```
└─ com.fs.party-starter-logging [boot] [party-construction master]
  └─ src/main/java
    └─ com.fs.party.starter.logging
      └─ aspect
        └─ WebLogAspect.java
      └─ config
        └─ MyWebAppConfigurer.java
      └─ controller
        └─ HelloController.java
      └─ interceptor
        └─ MyInterceptor.java
      └─ Application.java
    └─ src/main/resources
```


2. *com.fs.party.depeds* 部分 (略.....)


```
└─ com.fs.party.depeds [boot] [party-construction master]
  └─ src/main/java
    └─ com.fs.party.depeds
      └─ annotation
      └─ base
      └─ constant
      └─ data
      └─ date
      └─ page
      └─ pojo
      └─ spring
      └─ utils
```

3. *com.fs.party.system* 部分

 > com.fs.party.system [boot] [party-construction master]

└─  src/main/java

└─  com.fs.party

└─  system


└─  cfg

└─  controller


└─  api

└─  app

└─  client

└─  UserInfoController.java


└─  datasource


└─  exception

└─  pojo

└─  service

└─  utils

└─  Application.java

└─  src/test/java


4. com.fs.party.article 部分

 > com.fs.party.article [boot] [party-construction master]


└─  src/main/java


└─  com.fs.party

└─  article

└─  common


└─  controller.api


└─  BannerController.java


└─  LetterController.java


└─  NoticeController.java


└─  UeditorController.java


└─  dao.mybatis.party


└─  datasource


└─  exception


└─  pojo


└─  service


└─  impl


└─  BannerServiceImpl.java


└─  LetterServiceImpl.java


└─  NoticeServiceImpl.java


└─  SystemClientHystrix.java

└─  IBannerService.java

└─  ILetterService.java

└─  INoticeService.java

└─  ISystemClient.java

└─  Application.java

5. 项目依赖

● 父项目 (construction)

construction/pom.xml

Artifact

Group Id: com.fs.party
Artifact Id: construction
Version: 0.0.1-SNAPSHOT
Packaging: pom

Parent

Group Id: org.springframework.boot
Artifact Id: spring-boot-starter-parent
Version: 1.5.3.RELEASE
Relative Path:

Properties

project.build.sourceEncoding : UTF-8

Modules

com.fs.party.system
com.fs.party.meeting
com.fs.party.system.gateway
com.fs.party.meeting.gateway
com.fs.party.server
com.fs.party.depends
com-fs-party-starter-logging

Project

Name: HST-Construction
URL:
Description: 好视通党建平台

Inception:

Organization
SCM
Issue Management
Continuous Integration

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

construction/pom.xml

Dependencies

com.fs.party.depends : 0.0.1-SNAPSHOT
com-fs-party-starter-logging : 0.0.1-SNAPSHOT

Dependency Management

spring-cloud-dependencies : Dalston.RELEASE : pom [import]

To manage your transitive dependency exclusions, please use the [Dependency Hierarchy](#) page.

● 子项目 (com-fs-party-starter-logging)

construction/pom.xml | com-fs-party-starter-logging/pom.xml

Dependencies

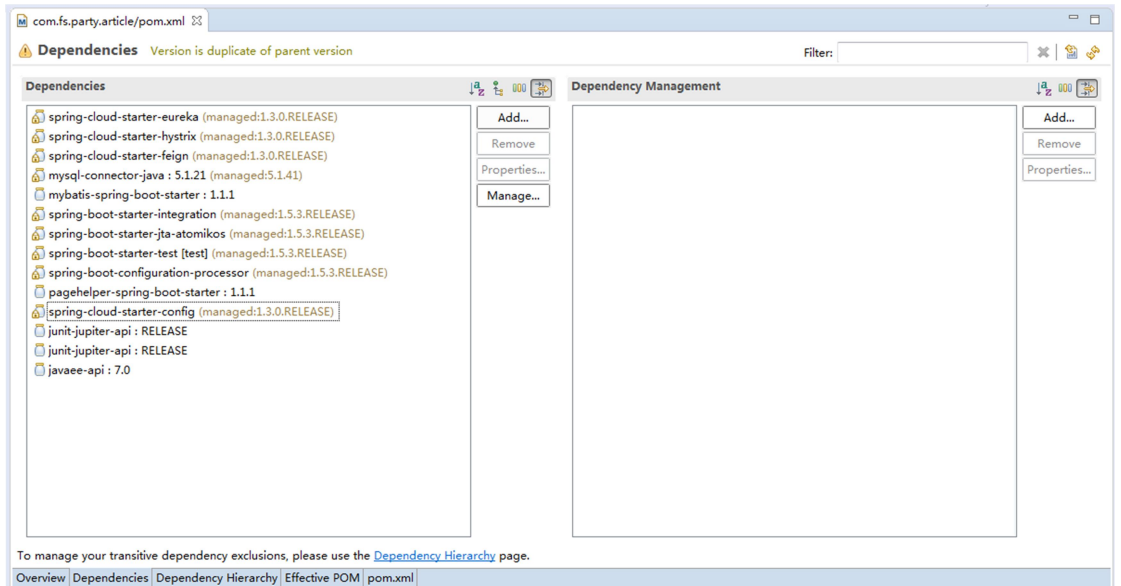
spring-boot-starter (managed:1.5.3.RELEASE)
spring-boot-starter-web (managed:1.5.3.RELEASE)
spring-boot-starter-aop (managed:1.5.3.RELEASE)
spring-boot-starter-test [test] (managed:1.5.3.RELEASE)
com.fs.party.depends : 0.0.1-SNAPSHOT

Dependency Management

To manage your transitive dependency exclusions, please use the [Dependency Hierarchy](#) page.

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

- 子项目 (com.fs.party.article)



五、总结

1. 现阶段总体框架已经完成，对于项目实施过程中暂时应用于单点部署形式，后续会对多点部署持续改进。
2. 对于网络流量总入口控制还未做出合适处理，需要找到相应的实现方案来解决。